Introduction aux expressions régulières

$Id: re.fodp 458 2012-04-30 16:19:25Z jaclin $

Jacquelin Charbonnel

ANF Mathrice - Angers, mai 2012

RE POSIX

jockers

- un caractère quelconque
  - t.t. : …toto…, …tutu…, …tati…

quantificateurs

* 0 ou plusieurs fois ce qui précède

- abc* : …ab…, …abc…, …abcc…, …abccc…

? 0 ou 1 fois ce qui précède

- abc? : …ab… ou …abc…

\+ au moins 1 fois ce qui précède

- abc+ : …abc…, …abcc…, …abccc…

échappement

\ ce qui suit est un littéral

- abc* : …abc*…
- * : …, …\…, …\\…, …\\\…, etc

assertions

^ début de ligne

- ^abc : abc…

$ fin de ligne

- abc$: abc

alternatives

| ou

- ^(Chapter|Section) [1-9]$

  ^Chapter [1-9]$ ou ^Section [1-9]$

[…] 1 caractère parmi un ensemble

- [abc] : …a… ou …b… ou …c…
- [a-zA-Z] : une lettre
- [a\-d] : …a… ou …-… ou …d…

[^…] 1 caractère hors d'un ensemble

- [^0-9] : tout sauf un chiffre

exemples

- ^trax$
  - trax
- ^[st]ac
  - sac… ou tac…
- peu[xt]?
  - …peu…, …peux… ou …peut…
- a[ou]+
  - …aou…, …ao…, …auuu…, …aououuuoou…
- [cC](hat|hien)
  - …chat…, …Chat…, …chien…, …Chien…

RE POSIX étendues

quantificateurs

- {n} n fois
- ab{2} : …abab…

- {n,m} k fois, k compris entre n et m
- ab{2,3} : …abab…, …ababab…
- ab{,3} : …, …ab…, …abab…, …ababab…
- ab{2,} : …abab…, …ababab…, etc.

groupement et capture

(…) groupe et capture

- (e|ae)quo : …equo… ou …aequo…

suivant l'appli (vim, emacs), il peut falloir les échapper :

- \(…\)
- \{…\}

les quantificateurs sont gourmands :

- X(.*) : capture ce qui suit le premier X
  - équivalent à X(.*)$
- ^(.*)X : capture ce qui précède le dernier X
- ^([^X]*)X : capture ce qui précède le premier X

Quelques extensions de Perl (PCRE)

jockers (classes de caractères)

\d	[0-9]
\D	[^0-9]
\s	1 séparateur blanc
\S	1 non séparateur blanc
\w	[A-Za-z0-9_àâéèêëîïôöùûüÿÿ]
\W	[^A-Za-z0-9_àâéèêëîïôöùûüÿÿ]

assertions

\b début ou fin de mot

\bsimples?\b

Voici simplement un simple exemple, très simple, mais pas simplet sur les assertions PCRE pourtant pas si simples!

quantificateurs non gourmands

*? * non gourmand

+? + non gourmand

{n,m}? {n,m} non gourmand

- ^(.*?)X : ce qui précède le premier X est capturé

parenthèses non capturantes

- (?:…) (…) non capturantes
  - (?:le|la|de|du)\s(\w+)
  - capture seulement le mot suivant l'article

quelques modificateurs

- print if /zut/i ; # zut Zut ZUT zUt zuT ZUt zuT
- $s =~ s/a/A/g ; # tous les a deviennent A
- $s =~ s/(\d+)/fct($1)/e ;

remplace le premier nombre par le retour de la fonction fct appliqué à ce nombre

option multi-lignes

soit : $s = "ligne 1\nligne 2" ;

^ et $ par rapport à la chaîne

$s =~ /ligne 1$/ est faux

avec /m, ^ et $ par rapport à chaque ligne

$s =~ /ligne 1$/m est vrai

option simple ligne

$s = "ligne 1\nligne 2" ;

. ne matche pas \n

$s =~ /ligne 1.ligne2/ est faux

avec /s, . matche \n

$s =~ /ligne 1.ligne2/ est vrai

variables dans les motifs

```
$s = "toto" ;
        if (/$s/) ... # équivalent à if (/toto/) ...

$s = "(-;" ;
if (/$s/) ...
        # incorrect, equivalent à : if (/(-;/)... -> pb de ()

quotemeta($s) -> \(-;
```